SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

HIGH SPEED TURBO CODES DECODER FOR 3G USING PIPELINED SISO LOG-MAP DECODERS ARCHITECTURE

Cross Reference to Related Applications

Referenced-applications

This patent is based upon the development of IP Core ASIC products for 3G-WDMA and 3G-CDMA2000 wireless communications applications by IComm Technologies, Inc. since early 1998. Other U.S. Patent References: 6,023,783 02/2000 Divsalar et al. 5,233,629 08/1993 Paik et al. 5,446,747 08/1995 Berrou 5,721,745 02/1998 Hladik et al. 5,729,560 03/1998 Hagenauer et al. 5,734,962 03/1998 Hladik et al.

Background of Invention

Field of the Invention

[0001] This invention relates to Baseband Processor and Error-Correction Codes for 3G Wireless Mobile Communications; and more particularly, to a very high speed Turbo Codes Decoder using pipelined Log-MAP decoders method for 3G CDMA2000 and 3G WCDMA.

Description of Prior Art

[0002]

A Turbo Codes Decoder is an important baseband processor of the digital wireless communication Receiver, which was used to reconstruct the corrupted and noisy received data and to improve BER (bit-error-rate) throughput. The FIGURE 1. shows an example of a 3G Receiver with a Turbo Codes Decoder 13 which decodes data from the Demodulator 11 and De-mapping 12 modules, and sends decoded data to the MAC layer 14. The FIGURE 2. shows an example of an 8-PSK constellation points 21 produced by the Demodulator module 11. The De-mapping 12 module uses the 8-PSK constellation points 21 to convert into binary data 22 and send to the Turbo Codes Decoder 13. The data 22 is then decoded and reconstructed by the Turbo Codes Decoder 13 and send to the MAC layer 14.

[0003] A most widely used FEC is the Viterbi Algorithm Decoder in both wired and wireless application. The drawback is that it would requires a long waiting for decisions until the whole sequence has been received. A delay of six time the memory of the received data is required for decoding. One of the more effective FEC, with higher complexity, a MAP algorithm used to decode received message has comprised the steps of very computational complex, requiring many multiplications and additions per bit to compute the posteriori probability. The major difficulty with the use of MAP algorithm has been the implementation in semiconductor ASIC devices, the complexity the multiplications and additions which will slow down the decoding process and reducing the throughput data rates. Furthermore, even under the best conditions, each multiplication will be used in the MAP algorithm, that would create a large circuits in the ASIC. The result is costly, and low performance in bit rates throughput.

[0004] Recently introduced by the 3GPP organization a new class of codes using parallel concatenated codes as shown in FIGURE 3. similar to the reference and patents to Berrou describing a basic parallel concatenated codes which is known as Turbo Codes. An example of the Turbo Codes with 8-states and rate 1/3 is shown in FIGURE 3. , data enters the two systematic encoders 31 33 separated by an interleaver 32. An output codeword consists of the source data bit followed by the parity check bits of the two encoders. Another patent US6023783 by Divsalar and Pollara describes a more improved encoding method than Berrou and some basic mathematical concepts of parallel concatenated codes. However, patents by Berrou, the US6023783, and others only describe the basic concept of parallel concatenated codes using mathematical equations which are good for research in deep space communications and other government projects but are not feasible, economical, and practical for consumers. The encoding of data is simple and can be easily implemented with a few xor and flip-flop logic gates. But the decoding the Turbo Codes is much more difficult to implement in ASIC or software. The prior arts describe briefly the implementation of the Turbo Codes Decoder which are mostly for deep space communications and requires much more hardware, powers and costs.

[0005] All the prior arts of Turbo Codes fail to achieve a simpler method for a Turbo Codes Decoder as it is required and desired for 3G cellular phones and 3G personal communication devices including high speed data throughput, low power consumption, lower costs, limited bandwidth, and limited power transmitter in noisy environment.

Summary of Invention

[0006] The present invention concentrates only on the Turbo Codes Decoder to implement a more efficient, practical and simpler method to achieve the requirements for 3G cellular phones and 3G personal communication devices including higher speed data throughput, lower power consumptions, lower costs, and simpler implementation in ASIC or software. The present invention encompasses improved and simplified Turbo Codes Decoder method and apparatus to deliver higher speed and lower power especially for 3G applications. Our Turbo Codes Decoder utilizes two pipelined and serially concatenated SISO Log-MAP Decoders with Interleaver-Memory at the output of the first decoder and a De-interleaver-Memory at the second decoder. The two decoders function in a pipelined scheme; while the first decoder is decoding data in the de-interleaver-Memory, the second decoder performs decoding data in the interleaver-Memory, which produces a decoded output every clock cycle in results. Accordingly, several objects and advantages of our Turbo Codes Decoder are:

[0007] To deliver higher speed throughput and lower power consumption

[0008] To utilize SISO Log-MAP decoder for faster decoding and simplified implementation in ASICr with the use ofbinary adders for computation.

[0009] To perform re-iterative decoding of data back-and-forth between the two Log-MAP decoders in a pipelined scheme until a decision is made. In such pipelined scheme, a decoded output data is produced each clock cycle.

[0010] To improve higher performance in term of symbol error probability and low BER for 3G applications such as 3G W-CDMA, and 3G CDMA2000 operating at very high bit-rate up to 100Mbps in a low power noisy environment.

[0011] To utilize an simplified and improved architecture ofSISO Log-MAP decoder including branch-meric (BM) calculations module, recursive state-metric (SM) forward/backward calculations module, Log-MAP posteriori probability calc ulations module, and output decision module.

[0012] To reduce complexity of multiplier circuits in MAP algorithm by perform the entire MAP algorithm in Log domain with the uses of binary der circuits which are more suitable for ASIC implementation while still maintain a high level of performance output.

[0013] To design an improve Log-MAP Decoder using high level design language of VHDL which can be synthesized into custom ASIC and FPGA devices.

[0014] Still further objects and advantages will become apparent to one skill in the art from a consideration of the ensuing descriptions and accompanying drawings.

Brief Description of Drawings

[0015] FIGURE 1. is a typical 3G Receiver Functional Block Diagram which use Turbo Codes Decoder for error-correction.

[0016] FIGURE 2. is an example of an 8-PSK (QPSK) constellations of the receiving signals.

[0017] FIGURE 3. is a block diagram of the 8-states Parallel Concatenated Convolutional Codes.

[0018] FIGURE 4. is the Turbo Codes Decoder System Block Diagram showing Log-MAP Decoder, Interleavers, Input Shift registers and control logics.

[0019] FIGURE 5. is a block diagram of the Input Buffer Shift Registers.

[0020] FIGURE 5b. is the Soft Values Mapping ROM Table.

[0021] FIGURE 6. is the input Buffer Interface Timing.

[0022] FIGURE 7. is a block diagram of the 8-states SISO Log-MAP Decoder showing Branch Metric module, State Metric module, Log-MAP module, am State and Branch Memory modules.

[0023] FIGURE 8. is the 8-States Trellis Diagram of a SISO Log-MAP Decoder.

[0024] FIGURE 9. is a block diagram of the BRANCH METRIC COMPUTING module.

[0025] FIGURE 10a. is a block diagram of the Log-MAP computing for u=0.

[0026] FIGURE 10b. is a block diagram of the Log-MAP computing for u=1.

[0027] FIGURE 11. is a block diagram of the Log-MAP Compare & Select l maximumogic for each state.

[0028] FIGURE 12. is a block diagram of the Soft Decode module.

[0029] FIGURE 13. is a block diagram of the Computation of Forward Recursion of stSte-m Mric module (FACS).

[0030] FIGURE 14. is a block diagram of the Computation of Backward Recursion of st Ste-m Mric module (BACS).

[0031] FIGURE 15. showing FoState Metric rward computing of Trellis state transitions.

[0032] FIGURE 16. showing BaState Metric ckward computing of Trellis state transitions.

[0033] FIGURE 17. is a block diagram of the State Machine operations of Log-MAP Decoder.

[0034] FIGURE 18. is a block diagram of the BM dual-port Memory Module.

[0035] FIGURE 19. is a block diagram of the SM dual-port Memory Module.

[0036] FIGURE 20. is a block diagram of the Interleaver RAM Memory Memory Module.

[0037] FIGURE 21. is a block diagram of the De-Interleaver RAM Memory Module.

[0038] FIGURE 22. is a block diagram of the State Machine operations of the Turbo Codes Decoder.

[0039] FIGURE 23. is a block diagram of the Iterative decoding feedback control Mux.

Detailed Description

Turbo Codes Decoder

[0040] As shown in FIGURE. 4, a Turbo Codes Decoder has two concatenated ~~Lo~~SISO lo g-MAP Decoders A 42 and B 44 connected in a feedback loop with Interleaver Memory 43 and De-Interleaver Memory 45 in between. An input Buffer 41, shown in details FIGURE. 5, has one serial-to-par(S/P) converter 51, and three shift registers 52 of length N for block decoding. A control logic module (CLSM) 47, shown in FIGURE. 4, consists of various state-machines which in turn control all the operations of the Turbo Codes Decoder. The hard-decoder module 46 outputs the final decoded data. Signals R2, R1, R0 are the received data shifted out from the Shift Registers. Signal XO1, and XO2 are the output soft decision of the Log-MAP Decoders A 42 and B 44 respectively, which are stored in the Interleaver Memory 43 and De-Interleaver Memory 45 module. Signal Z2 and Z1 are the output of the Interleaver Memory 43 and De-Interleaver Memory 45 where the Z2 is feed into Log-MAP decoder B 44 , and Z1 is feedback into Log-MAP decoder A 42 for iterative decoding.

[0041] In accordance with the invention, the Turbo Codes Decoder decodes an 8-state Parallel Concatenated Convolutional Code (PCCC), with coding rate 1/3, constraint length K=4, using L SISOog- MAP (Maximum a Posteriori) Decoders in pipeline. The Turbo Codes Decoder can also decode a 16-states or more PCCC with different code rates.

[0042] As shown in FIGURE 1. the Turbo Codes Decoder functions effectively as follows:

[0043] Serial received data are shifted into 3 Shift Registers to produce R0, R1, and R2 data sequence.

[0044] The soft value module converts the input data R0, R1, and R2 into 3-bit quantization soft-values according to TABLE 1.

[0045] When a block of N input data is received, the Turbo Decoder starts the Log-MAP Decoder A to decode the N input bits based on the soft-values of R0 and R1, then stores the outputs in the Interleaver Memory.

[0046] Next, the Turbo Decoder starts the Log-MAP Decoder B to decode the N input bits based on the soft-values of R2 and Z2, then store the outputs in the De-Interleaver Memory.

[0047] The Turbo Decoder will now do the iterative decoding for L number of times. The Log-MAP Decoder A now uses the signals Z1 and R1 as inputs. The Log-MAP Decoder B uses the sigZ2 and R2 as inputs.

[0048] When the iterative decoding sequences are done, the Turbo Decoder starts the hard-decision operations to compute and produce soft-decision outputs.

Input Buffer Shift Registers

[0049] As shown in FIGURE. 5., the Turbo Codes Input Buffer has a serial-to-parallel (S/P) converter 51, and three Shift Registers 52 of N bits to store each block of N input data. A 3-bit Serial- to Parallel (S/P) converter 51 converts input data into 3 serial data streams which are then shifted into the corresponding shift registers 52. As shown in FIGURE. 6. is the timing interface with the input buffer from the external hosts or the Demodulator/De-mapping 12. A bit clock (BCLK) in conjunction with the frame sync (RSYNC) are used to shift data into the S/P converter 51 when the RSYNC is active (HIGH).

[0050] As shown in FIGURE 5b., each input data bit R0, R1, and R2 entering into the Log-MAP Decoder are assigned a soft-value of L-bit quantization as shown in the following TABLE 1. The same soft values are used as the threshold for the final hard code data.

[t2]

TABLE 1. Soft values

Soft Values L-bit	*Input data Bit*
.....011	0
.....101	1

SISO Log-MAP Decoder

[0051] As shown in FIGURE 7., an SISO Log-MAP Decoder42 44 comprises of a Branch Metric (BM) computation module 71, a State Metric (SM) computation module 72, a Log-MAP computation module 73, a BM Memory module 74, a SM Memory module 75, and a Control Logic State Machine module 76. Soft-values inputs enter the Branch Metric (BM) computation module 71, where Euclidean distance is calculated for each branch, the output branch metrics are stored in the BM Memory module 74. The State Metric (SM) computation module 72 reads branch metrics from the BM Memory 74 and compute the state metric for each state, the output state-metrics are stored in the SM Memory module 75. The Log-MAP computation module 73 reads both branch- metrics and state-metrics from BM memory 74 and SM memory 75 modules to compute the Log Maximum a Posteriori probability and produce soft-decision output. The Control Logic State- machine module 76 provides the overall operations of the decoding process.

[0052] As shown in FIGURE 7., the Log-MAP Decoder 42 44 functions effectively as follows:

[0053] The Log-MAP Decoder 42 44 reads each soft-values (SD) data pair input, then computes branch- metric (BM) values for all 16 paths in the Turbo Codes Trellis 85 as shown in FIGURE 8., then stores all 16 BM data into BM Memory 74. It repeats computing BM values for each input data until all N samples are calculated and stored in BM Memory 74.

[0054]

The Log-MAP Decoder 42 44 reads BM values from BM Memory 74 and SM values from SM Memory 75, and computes the forward state-metric (SM) for all 8 states in the Trellis 85 as shown in FIGURE 8., then store all 8 forward SM data into SM Memory 75. It repeats computing forward SM values for each input data until all N samples are calculated and stored in SM Memory 75.

[0055] The Log-MAP Decoder 42 44 reads BM values from BM Memory 74 and SM values from SM Memory 75, and computes the backward state-metric (SM) for all 8 states in the Trellis 85 as shown in FIGURE 8., then store all 8 backward SM data into the SM Memory 75. It repeats computing backward SM values for each input data until all N samples are calculated and stored in SM Memory 75.

[0056] The Log-MAP Decoder 42 44 then computed Log-MAP posteriori probability for u=0 and u=1 using BM values and SM values from BM Memory 74 and SM Memory 75. It repeats computing Log-MAP posteriori probability for each input data until all N samples are calculated. The Log-MAP Decoder then decodes data by making soft decision based on the posteriori probability for each stage and produce soft-decision output, until all N inputs are decoded.

Branch Metric Computation module

[0057] The Branch Metric (BM) computation module 71 computes the Euclidean distance for each branch in the 8-states Trellis 85 as shown in the FIGURE 8. based on the following equations:

[0058] Local Euclidean distances values = SD0*G0 + SD1*G1

[0059] The SD0 and SD1 are soft-values from TABLE 1., G0 and G1 are the expected input for each path in the Trellis 85. G0 and G1 are coded as signed antipodal values, meaning that 0 corresponds to +1 and 1 corresponds to -1. Therefore, the local Euclidean distances for each path in the Trellis 85 are computed by the following equations:

[0060] M1 = SD0 + SD1

[0061] M2 = - M1

[0062] M3 = M2

[0063] M4 = M1

[0064] M5 = - SD0 + SD1

[0065] M6 = - M5

[0066] M7 = M6

[0067] M8 = M5

[0068] M9 = M6

[0069] M10 = M5

[0070] M11 = M5

[0071] M12 = M6

[0072] M13 = M2

[0073] M14 = M1

[0074] M15 = M1

[0075] M16 = M2

[0076] As shown in FIGURE 9., the Branch Metric Computing module comprise of one L-bit Adder 91, one L-bit Subtracter 92, and a 2'complemeter 93. It computes the Euclidean distances for path M1 and M5. Path M2 is 2'complement of path M1. Path M6 is 2'complement of M5. Path M3 is the same path M2, path M4 is the same as path M1, path M7 is the same as path M6, path M8 is the same as path M5, path M9 is the same as path M6, path M10 is the same as path M5, path M11 is the same as path M5, path M12 is the same as path M6, path M13 is the same as path M2, path M14 is the same as path M1, path M15 is the same as path M1, and path M16 is the same as path M2.

State Metric Computing module

[0077] The State Metric Computing module 72 calculates the probability A(k) of each state transition in forward recursion and the probability B(k) in backward recursion. FIGURE 13. shows the implementation of state-metric in forward recursion with Add-Compare-Select (ACS) logic, and FIGURE 14. shows the implementation of state-metric in backward recursion with Add-Compare-Select (ACS) logic. The calculations are performed at each node in the Turbo Codes Trellis 85 (FIGURE 8) in both forward and backward recursion. The FIGURE 15. shows the forward state transitions in the Turbo Codes Trellis 85 (FIGURE 8), and FIGURE 16 . show the backward state transitions in the Turbo Codes Trellis 85 (FIGURE 8). Each node in the Trellis 85 as shown in FIGURE 8. has two entering paths: one-path 84 and zero-path 83 from the two nodes in the previous stage.

[0078] The ACS logic comprises of an Adder 132, an Adder 134, a Comparator 131, and a Multiplexer 133. In the forward recursion, the Adder 132 computes the sum of the branch metric and state metric in the one-path 84 from the state s(k-1) of previous stage (k-1). The Adder 134 computes the sum of the branch metric and state metric in the zero-path 83 from the state (k-1) of previous stage (k-1). The Comparator 131 compares the two sums and the Mulitplexer 133 selects the larger sum for the state s(k) of current stage (k). In the backward recursion, the Adder 142 computes the sum of the branch metric and state metric in the one-path 84 from the state s(j+1) of previous stage (J+1). The Adder 144 computes the sum of the branch metric and state metric in the zero-path 83 from the state s(j+1) of previous stage (J+1). The Comparator 141 compares the two sums and the Mulitplexer 143 selects the larger sum for the state s(j) of current stage (j).

[0079] The Equations for the ACS are shown below:

[0080] A(k) = MAX [(bm0 + sm0(k-1)), (bm1 + sm1(k-1)]

[0081] B(j) = MAX [(bm0 + sm0(j+1)), (bm1 + sm1(j+1)]

[0082] Time (k-1) is the previous stage of (k) in forward recursion as shown in FIGURE 15., and time (j+1) is the previous stage of (j) in backward recursion as shown in FIGURE 16.

Log-MAP computing module

[0083] The Log-MAP computing module calculates the posteriori probability for u=0 and u=1, for each path entering each state in the Turbo Codes Trellis 85 corresponding to u=0 and u=1 or referred as zero-path 83 and one-path 84. The accumulated probabilities are compared and selected the u with larger probability. The soft-decision are made based on the final probability selected for each bit. FIGURE 10a. shows the implementation for calculating the posteriori probability for u=0. FIGURE 10b. shows the implementation for calculate the posteriori probability for u=1. FIGURE 11. shows the implementation of compare-and-select the u with larger probability. FIGURE 12. shows the implementation of the soft-decode compare logic to produce output bits based on the posteriori probability of u = 0 and u = 1. The equations for calculation the accumulated probabilities for each state and compare-and-select are shown below:

[0084] sum_s00 = sm0i + bm1 + sm0j

[0085] sum_s01 = sm3i + bm7 + sm1j

[0086] sum_s02 = sm4i + bm9 + sm2j

[0087] sum_s03 = sm7i + bm15 + sm3j

[0088] sum_s04 = sm1i + bm4 + sm4j

[0089] sum_s05 = sm2i + bm6 + sm5j

[0090] sum_s06 = sm5i + bm12 + sm6j

[0091] sum_s07 = sm6i + bm14 + sm7j

[0092] sum_s10 = sm1i + bm3 + sm0j

[0093] sum_s11 = sm2i + bm5 + sm1j

[0094] sum_s12 = sm5i + bm11 + sm2j

[0095] sum_s13 = sm6i + bm13 + sm3j

[0096] sum_s14 = sm0i + bm2 + sm4j

[0097] sum_s15 = sm3i + bm8 + sm5j

[0098] sum_s16 = sm4i + bm10 + sm6j

[0099] sum_s17 = sm7i + bm16 + sm7j

[0100] s00sum = MAX[sum_s00, 0]

[0101] s01sum = MAX[sum_s01 , s00sum]

[0102] s02sum = MAX[sum_s02 , s01sum]

[0103] s03sum = MAX[sum_s03 , s02sum]

[0104] s04sum = MAX[sum_s04 , s03sum]

[0105] s05sum = MAX[sum_s05 , s04sum]

[0106] s06sum = MAX[sum_s06 , s05sum]

[0107] s07sum = MAX[sum_s07 , s06sum]

[0108] s10sum = MAX[sum_s10 , 0]

[0109] s11sum = MAX[sum_s11 , s10sum]

[0110] s12sum = MAX[sum_s12 , s11sum]

[0111] s13sum = MAX[sum_s13 , s12sum]

[0112] s14sum = MAX[sum_s14 , s13sum]

[0113] s15sum = MAX[sum_s15 , s14sum]

[0114] s16sum = MAX[sum_s16 , s15sum]

[0115] s17sum = MAX[sum_s17 , s16sum]

Control Logics – State Machine (CLSM) Module

[0116] As shown in FIGURE 7. the Control Logics module controls the overall operations of the Log- MAP Decoder. The control logic state machine 171, referred as CLSM, is shown in FIGURE 17. The CLSM module 171 (FIGURE 17.) operates effectively as the followings. Initially, it stays in IDLE state 172. When the decoder is enable, the CLSM transitions to CALC-BM state 173, it then starts the Branch Metric (BM) module operations and monitor for completion. When Branch Metric calculations are done, referred as bm-done the CLSM transitions to CALC-FWD-SM state 174, it then tarts the State Metric module (SM) in forward recursion operation. When the forward SM state metric calculations are done, referred as fwd-sm, the CLSM transitions to CALC-BWD-SM state 175, it then starts the State Metric module (SM) in backward recursion operations. When backward SM state metric calculations are done, referred as bwd-sm-done the CLSM transitions to CALC-Log-MAP state 176, it then starts the Log-MAP computation module to calculate the maximum a posteriori probability to produce soft decode output. When Log-MAP calculations are done, referred as log-map-done, it transitions back to IDLE state 172.

BM Memory and SM Memory

[0117] The Branch-Metric Memory 74 and the State-Metric Memory 75 are shown in FIGURE 7. as the data storage components for BM module 71 and SM module 72. The Branch Metric Memory module is a dual-port RAM contains M-bit of N memory locations as shown in FIGURE 18. The State Metric Memory module is a dual-port RAM contains K-bit of N memory locations as shown in FIGURE 19. Data can be written into one port while reading at the other port.

Interleaver Memory and De-interleaver Memory

[0118] As shown in FIGURE 4., the Interleaver Memory 43 stores data for the first decoder A 42, and De-interleaver memory 45 stores data for the second decoder B 44. In an iterative pipelined decoding, the decoder A 42 reads data from De-interleaver memory 45 and writes results data into Interleaver memory 43, the decoder B 44 reads data from Interleaver memory 43 and write results into De-interleaver memory 45.

[0119] As shown in FIGURE 20., the Interleaver memory 43 comprises of an Interleaver module 201 and a dual-port RAM 202 contains M-bit of N memory locations. The Interleaver is a Turbo code internal interleaver as defined by 3GPP standard ETSI TS 125 222 V3.2.1 (2000-05). The Interleaver permutes the address input port A for all write operations into dual-port RAM module. Reading data from output port B are done with normal address input.

[0120] As shown in FIGURE 21., the De-Interleaver memory 45 comprises of an De-Interleaver module 211 and a dual-port RAM 212 contains M-bit of N memory locations. The De-Interleaver is a Turbo code internal interleaver as defined by 3GPP standard ETSI TS 125 222 V3.2.1 (2000-05). The De-Interleaver permutes the address input port A for all write operations into dual-port RAM module. Reading data from output port B are done with normal address input.

Turbo Codes Decoder Control Logics - State Machine (TDCLSM)

[0121] As shown in FIGURE 4. the Turbo Decoder Control Logics module 47, referred as TDCLSM, controls the overall operations of the Turbo Codes Decoder. The state-machine is shown in the FIGURE 22. Initially, the TDCLSM 47 stays in IDLE state 221. When decoder enable signal is active, the TDCLSM 47 transitions to INPUT states 222. The TDCLSM 47 starts the input shifting operations until the input buffer 41 is full indicated by input-ready signal. Then, the TDCLSM 47 transitions to Log-MAP A state 223 and starts the operations of the Log-MAP Decoder A 42. When the Log-MAP A 42 is done, the TDCLSM 47 transitions to the Log-MAP B state 224 and starts the Log-MAP Decoder B 44. When Log-MAP B 44 is done, the TDCLSM 47 starts the iterative decoding for J number of times. When the iterative decoding sequences are done, the TDCLSM 47 transitions to HARD-DEC state 225 and produces the hard-decode outputs. Then the TDCLSM 47 transitions to the INPUT state to start decoding another block of data.

Iterative Decoding

[0122] Turbo Codes decoder performs iterative decoding L times by feeding back the output Z1 of the second Log-MAP decoder B into the first Log-MAP decoder A, before making decision for hard- decoding output. As shown in FIGURE 13., the Counter 233 count the preset number L times, the Multiplexer select the input for the Decoder A. When Counter is zero, the Mux selects the input R1, else it selects the recursive input Z1 from Decoder B.